D0 note 2958
CMS TN/96-062

Finding Tracks

David Adams
Rice University

May 14, 1996

Abstract: A high level design for pattern recognition in the central tracking system.

1 Introduction

One of the most challenging problems in the reduction of data from modern hadron colliders is pattern recognition in the central tracking system. These colliders are characterized by both high energy (multi-TeV) and high luminosity ($10^{32\text{-}34}$ cm^{-2}s^{-1}). Each event produces many charged particles per unit of rapidity and many events may occur in a single beam crossing. The central tracker is subjected to a high density of tracks stressing the pattern recognition whose performance is a rather sensitive function of this track density.

Here we outline the major components of an algorithm to find tracks under these circumstances. We demand that this algorithm make efficient use of computing resources and that it be tunable, *i.e.*, there are parameters which can be adjusted to trade off between track-finding performance and computing resources. The latter is important because it appears likely there will not be sufficient computing resources to carry out "complete" reconstruction for all events.

The algorithm described here is road-following with a Kalman filter update. It is iterative: a hit is added to an existing track to create a new track. The new track is rejected if it is not suitable (*e.g.* if its chi-square is too large). The problem is then reduced to selecting candidate hits, adding them to the existing track and checking the quality of the new track. While it is essential that the last two steps be implemented in a computationally efficient manner, the hit selection offers the greatest opportunity for improving performance by reducing the number of candidate hits.

We begin by describing what is meant by track finding and then proceed to define each of the elements of the algorithm.

2 The Problem

As charged particles pass through the central tracker, they deposit energy in various detector elements. The resulting signals are recorded and later analyzed to infer the positions of the particles. We call these measurements hits. There are many tracks and each of these pass through many such elements. The problem of track finding is to assign hits to tracks, *i.e.* determine the ordered list of hits associated with each charged particle. A closely related but different problem is track fitting: determining the kinematic track parameters based on the hits.

3 Surfaces

We begin by defining the geometrical framework of the problem. The detector is described as a series of non-intersecting surfaces and all hits are assigned to one of these surfaces. These surfaces are finite. Examples include a rectangle and part of a cylinder. We do not preclude the possibility of associating a normal dimension with a surface so that it actually corresponds to a thin volume.

Surfaces allows us to restrict the range of hits which must be searched when looking for the next hit on a track. The hits are assumed to be distinct although they may overlap. A track may pick up at most one hit when it crosses a surface and two (or more) tracks may share a hit. A track may cross a surface without picking up a hit to allow for detector gaps and inefficiencies. We call these missed surfaces. Shared hits and missed surfaces provide powerful criteria for identifying false tracks.

These logical surfaces will naturally correspond closely to actual detector elements. Note however that a surface might include multiple elements or one detector element might be divided into multiple surfaces.

4 Paths

Primarily a path is an ordered list of surfaces crossed by a track. It might also explicitly indicate (in order) surfaces which are not crossed. In general a path may be extended by adding another surface. Otherwise it is called a complete path. Paths serve two purposes: to group tracks and to provide instructions for track finding.

The grouping of tracks by paths allows us to assign path-specific track properties. Some of these will be described in the following sections. Here we mention a few examples. One example is starting kinematical values for the track fit. Perhaps most important are the maximum allowed chi-square and maximum number of missed surfaces used to reject bad tracks. Another example might be the order in which surfaces are added to a track. All of these can be tuned much more precisely if they are path-specific rather than global values. It is also useful to monitor the track-finding by noting the computing resources consumed by particular paths or sets of paths.

A list of complete paths serves as a map for track finding. The use of computing resources is optimized by searching only for tracks consistent with the specified map. Paths which produce relatively few good tracks for a given amount of computing resources or which cover a less interesting region may be excluded. They may then be added later without repeating the track finding for the paths in the initial map.

5 Hits

Raw detector data (strips, wires, fibers, ADC and TDC information,...) are reduced to form the hits which are used as input for track-finding. Typically, these are either one- or two-dimensional measurements indicating where the track crossed the surface. However, there is no reason these cannot be generalized to include the direction or any combination. The only requirement is that there must be a known procedure for predicting the hit parameters from the track parameters at the surface.

The data associated with a hit include constant parameters such as the surface normal coordinate and mixing angles (*e.g.* a stereo angle) and the measured parameters and their error matrix. Specifying the normal coordinate allows the hit to displaced from its surface.

6 Tracks

The essence of a track is an ordered list of hits. We add to this a path (or list of paths). The path (or paths) must be consistent with the list of hits. A track may be fitted to obtain the most likely set of kinematical track parameters, their error matrix and a fit chi-square. These are used to discriminate real tracks from ghosts. We will not discuss the many options for specifying the track parameters except to note that six parameters (three position, two direction and a curvature) are necessary to specify a position on a track and five to specify the track itself.

6.1 Propagation

In order to predict a hit from a track, it is first necessary to propagate that track (more precisely its parameters and error matrix) to the associated surface. This, of course, requires knowledge of the magnetic field. If the material distribution is also known, then it is also possible to account for multiple scattering, dE/dx losses and fluctuations in the latter.

6.2 Hit Selection

One a track has been propagated to a surface, it is desirable to fit only those hits which are "nearby" rather than trying all hits on the surface. It is strongly recommended that a procedure to identify such hits be implemented.

6.3 Fitting

We assume fitting is done with a Kalman filter update. Here we provide only a brief description of the input and output without discussing any of the mathematics behind this well-known technique. The Kalman filter allows us to take the track parameters and their error matrix and add the information from a hit to generate a new set of track parameters and error matrix. It is necessary to propagate the track to the surface associated with the hit before fitting.

We have implicitly assumed the track finding proceeds in one direction (*e.g.* inside to out or outside to in). It is possible to add points in between but the procedure is much more complicated and requires that additional information be stored for each hit.

6.4 Starting Values

The propagation of tracks is nonlinear as may be the prediction of hits from tracks. This makes the fitting procedure nonlinear and can result in sensitivity to the initial values for the track parameters and their errors. There must be a procedure (possibly path-dependent) to choose these parameters before the first hit is added. It may be desirable to refit all points for the first few points until the starting values have stabilized.

6.5 Discrimination

Once the paths have been selected, the procedure for adding hits is straightforward. However, when the track density is high, the number of tracks can increase very rapidly and it is necessary to start identifying and rejecting ghost tracks early in the track finding. Tracks can be rejected if there chi-square is too large, they have too many missed surfaces, they fall outside the kinematical range of interest, their fit is inconsistent with their paths, *etc.*

It is also possible to reject tracks based on hits shared with other tracks but this require that all tracks (or all tracks with a particular path) be fit up to the same surface. This has implications for memory usage, CPU time and parallelism and is probably best left for the end or used sparingly.

7 Conclusions

We have outlined an algorithm for finding tracks in a high-energy, high-luminosity hadron collider. We advocate the use of the Kalman filter as it largely decouples the track propagation from the track fitting. This allows a straightforward description of multiple scattering and dE/dx losses. The Kalman filter is likely to be a central part of any modern track fitting code.

We also emphasize the importance of making early decisions to minimize computational requirements. Hits should be checked for closeness before attempting to add them to a track. Track discrimination should be carried out from the very beginning. These are important but can probably be added in later if they are not included in the initial design.

The concept of paths and their usage to tailor various parameters and provide a map for track finding is one we wish to emphasize. It appears unlikely there will ever be sufficient computing resources for "complete" reconstruction. Paths provide the programmer with finer tuning of discrimination and more options for resource allocation. We recommend this feature appear explicitly in the initial design to provide easy access to these advantages.

This description is deliberately missing many details. It is only intended to serve as a very high-level design. The appendices provide a few comments about implementation.

Appendices

A1 Object Oriented Design

The above discussion remains valid with any programming style but it is worthwhile connecting it to the object oriented methodology because of its current popularity and its natural application here. Each of the major sections (Surfaces, Paths, Hits and Tracks) can be naturally associated with a class. The data associated with each can be found in the associated text and the methods associated with tracks correspond closely to the sub-headings of that section.

Paths could be implemented as a linked list of surfaces with a pointer up to the previous path surface and pointers down to all possible next surfaces. Data associated with each path surface would include discrimination parameters, consumed CPU resources, *etc.*

A2 Parallelism

With the above design, tracks are independent of one another and self-propagating up to the point where discrimination is based on shared hits. Thus it would be natural to keep one or more a stacks of tracks and let multiple processes add and remove tracks from the stack(s).